ALPHAMIC

THE MICROPROGRAMMING LANGUAGE FOR THE

HEWLETT-PACKARD HP3000

(PRODUCTION PROTO TYPE VERSION REVISED)

MARCH 28, 1972

7 June 1971	RBUS	SBUS	FCN.	SHIFT	STORE	SPEC.*	SKIP	MCU*	
00	·PL	CIR	TASL	LRZ	MASK	CCB	ZERO		00
01	SR	SPI	TASR	LLZ	IOA	CCPX	NZRO		01
02	Z	PADD	ROMX	SL1	IOD	SRØ	EVEN	d	02
03	MREG		ROMN	SR1	MREG	HALT	ODD	d	03
04	PADD	CPXI	JSB	RRZ	BUSL*	SIFG	NSME	d	04
05	RBUS·	MOD	CAND	RLZ	BSPØ*	SDFG	BIT6	d	05
06	X	CPX2	XOR	ROT	BUSH*	CTF	BIT8		06
07	XC	SWCH	AND	NOP	DATA	CF3	NOFL	d	07
10	RD	QDWN	DVSB		PUSH	INSR	CRRY		10
11	RC	IOA	UBNT		PL	DCSR	NCRY		11
12	RB	IOD	CADO		Z	INCN	POS	d	12
13	RA	MASK	SUBO		QUP	INCT	NEG	d	13
14	SPI	CTRL	JMP		SPI	HBF	F1	d	14
15	SPØ	CTRH	BNDT		SPØ	FHB	NF1	d	15
16	UBUS	UBUS	CAD		CTRL	CLIB	F2		16
17	NOP	SBUS	SUB		CTRH	LBF	NF2	d	17
20		P	PNLR+		P	SF2	SRZ		20
21		Q	PNLS+		Q	CF2	SRNZ		21
22		DB	ROMI		SM	CF1	SR4	d	22
23		SM	ROM+		DB	SF1	SRN4	d	23
24		STA	REPC+		STA	SCRY	INDR	RWAN·	24
25		SP3	REPN+		SP3	CCRY	SRL2	"	25
26		OPND	IOR		X	POPA	NPRY	"	26
27		CC	CTSD+		RAR	POP	SRL3	RWAN	27
30		RD	MPAD+		RD	SOV	RSB	CMD	30
31		RC	INCO+		RC	CLO	JLUI	CRL	31
32		RB	CRS+		RB	CCZ	TEST	NIR	32
33		RA	ADDO+		RA	CCL	CTRM·	RWN	33
34		DL	CTSS+		DL	CCG	F3	OPND	34
35		SP2	INC+		SP2	CCE	NEXT	RNWA	35
36		PB	RPTY+		PB	CCA	UNC	CWA	36
37		NOP	ADD+		NOP	NOP	NOP	RWA	37

* THESE STORE OPTIONS 'NOP' THE SPECIAL FIELD FUNCTIONS AND CAUSE THE FIELD TO BE USED FOR MCU OPTIONS.

† THESE FUNCTIONS CAUSE AN 'ADD'.

· THIS RWAN OPTION EMITTED BY μ ASSEMBLER

d DON'T CARE

PHYSICAL LAYOUT

0 1	2 3 4	5 6 7	8 9	10 11	12 13	14	15 16	17 18 19	20 21 22	23 24 25	26 27	28	29 30 31

S	STORE	REPN	COUNT	SHIFT	SPECIAL	R
		FUNCTION	SKIP			
		JMP, JSB		JUMP TARGET		
		any ROM	X ROM CONSTANT			

A BRIEF EXPLANATION OF THE MICROINSTRUCTION FIELDS

There are nine fields in the microinstruction:

The LABEL field may be any characters in columns 1-4 provided that the first character is not the blank character, an asterisk (*), a number sign (#), or a slash (/). The asterisk indicates a comment card which will appear on the listing but not affect the assembly. The number sign indicates the end of an ALPHAMIC assembly. The slash forces the next assembled word to reside in the location immediately following the slash. If no number is present after the slash, the next assembled word will be assembled to the start of the next 256 word sector boundary point.

The R-BUS field in columns 6-9 points to the register to be placed in the R-BUS register.

The S-BUS field in columns 11-14 points to the register to be placed in the S-BUS register.

The FUNCTION field in columns 16-19 gives the function that the arithmetic logic unit (ALU) is to perform on the two operands contained in the R-BUS and S-BUS registers or a special function.

The SHIFT field in columns 21-24 denotes how the information resulting from the ALU and placed on the T-BUS should be shifted and placed on the U-BUS.

The STORE field in columns 26-29 points to a register in which the contents of the U-BUS are to be stored.

The SPECIAL field in columns 31-34 has many varied uses which are best explained in that section of this document.

The SKIP field in column 36-39 denotes conditions of the CPU that logical decisions in the micro-program can be made.

Top of the Stack

The stack has a topmost element which is LOGICALLY the quantity A. Similarly, there is a LOGICAL quantity B, C, and D corresponding to the second, third, and fourth word of the stack, respectively. The LOGICAL quantities A, B, C, and D may be either in registers or in memory. This is determined by the SR register. If the SR register is 0 then none of the logical quantities A, B, C, or D are in registers but rather they are located in memory locations (SM), (SM-1), (SM-2), and (SM-3), respectively.

At all times however, there are four registers RA, RB, RC, and RD, which are named by a hardware naming device. In the micro-program the micro-options RA, RB, RC, and RD refer to the hardware named registers and NOT TO THE LOGICAL QUANTITIES A, B, C, and D. There is a correspondence however. For any of the LOGICAL quantities A, B, C, and D, the state of SR indicates where it is located by A, B, C, and D, the state of SR indicates where it is located by the following table:

SR	A	B	C	D
0	(SM)	(SM-1)	(SM-2)	(SM-3)
1	RA	(SM)	(SM-1)	(SM-2)
2	RA	RB	(SM)	(SM-1)
3	RA	RB	RB	(SM)
4	RA	RB	RB	RD

Note then that if SR = 1, B is in (SM) and if the micro-op RB is used, the contents of the register named RB will be affected, NOT THE LOGICAL quantity B.

The micro-store field instruction PUSH does three things:
1. Stores the output of the shifter into the register RD.
2. Increments the SR register.
3. Renames the registers so that
      N(RA) := RB, N(RB) := RC, N(RC) := RD, N(RD) := RA where
   N(RA) is read 'the name of RA'.

Similarly the micro-spec field instruction POP does two things:
1. Decrements the SR register
2. Renames the registers so that
      N(RA) := RD, N(RB) := RA, (N(RC) := RB, N(RD) := RC.

The micro-functions QUP, QDWN, MREG read and stores, are fully explained in the field descriptions and should be used very seldomly since the stack will be preadjusted in most cases.

INTERRUPTS

The following is a brief explanation of the hardware interrupt
information available to the microprocessor and hardware dependent sequences
that the microprocessor must execute to handle interrupts correctly. Interrupts
are detected via the interrupt status registers CPX1 and CPX2. CPX1 contains
all run state interrupts, that is, those that occur while the CPU is executing
instructions, and CPX2 contains all control panel interrupts and status information.
The special field option CCPX is used to control the information in the interrupt
registers and hardware dependent sequences.

Note that all interrupt bits in both CPX1 and CPX2 will cause a hardware jump
to ROM location 2 if a NEXT skip field option is executed and will cause the
TEST skip condition to be true.

Control of the interrupt bits is accomplished by the run-halt state.
In run state, all interrupt bits in CPX1 are allowed, while all interrupt bits
in CPX2 are held off. In halt state, front panel interrupts are allowed, while
all interrupt bits in CPX1 are held off. Note that this operation may be
overridden with control panel options.

Each interrupt bit in CPX1 except integer overflow is cleared by
executing a CCPX special option with the appropriate bit on the U-BUS. Note
that the clearing of the power fail inhibits all other interrupt bits in both
CPX1 and CPX2. All interrupt bits in CPX2 are cleared by executing a CCPX with
bit 15 of the U-BUS being a 1.* Note that run, execute switches, and single
instruction interrupts must be cleared before the execution of a NEXT. In
the case of run interrupt, the CCPX (15) enables the interrupt bits in CPX1 to
come through. A CCPX (15) after a cold load or cold dump interrupt enables the
run state. The interrupt stack and dispatcher flags are cleared by executing a
CCPX with the appropriate bits. CCPX (13) lights the system halt lamp on the
front panel, while CCPX (14) causes the CPU to freeze. These conditions are
irrevocable. CCPX (Ø) gates the contents of NIR register to CIR register.
The look-up table function may be disabled by executing CCPX(Ø) in a repeat loop.
The CPU timer bit (CPX1(4)) must be cleared after a CPU reset.

* Interrupt bits include CPX2(0:6) and CPX2(15).

TABLE 1

UBUS/CPX BIT ASSIGNEMENTS

BIT	CPX1	CCPX	CPX2
Ø	Integer Ovfl	NIR to CIR	Run FF
1	System Parity Error	Clear SPE FF	Cold Load
2	Addr. Parity Error	Clear APE FF	Single Inst
3	Data Parity Err	Clear DPE FF	Load Reg
4	CPU Timer	Clear Timer FF	Display Mem
5	BNDV	Clear BNDV FF	Load Mem
6	Module Viol	Clear MOD V FF	Execute SW
7	Module INTRP	Clear MOD INTFF	Incr. Addr
8	External Intrp	Clear EXTINT FF	Decr. Addr.
9	Console Intrp.	Reset Fnt. Pan. FF	Inhibit Auto Rest
1Ø	Power Fail	Turn off intrps	I/O Timer
11	Ø	Clear INT Stk Flg	INTRP Stk Flag
12	Ø	Clear Disp. Flag	Dispatch Flag
13	Ø	Set Error Light	Ø
14	Ø	uCode Halt	Ø
15	Ø	Reset Fnt. Pan. FF	Cold Dump

NOTES:

> CPX1 bits 2, 3, and 6 catch CPU — Memory errors only. I/O
>   errors are invisible.
>
> CCPX U9 clears general Intrp. Ckt only.
>
> CCPX U15 clears general intrp, sing. instr/exeq, and MCU CMP halt
>   ckts.
>
> CCPX U10, U13, U14 are irrevocable.

R-BUS Field

(blank)   Zero is placed in the R-BUS register.

MREG   SP1(14:15) are added to the contents of the namer register
               to get a temporary name. This is used to reference a memory
               element that happens to lie in the TOS. Register SP1(14:15)
               contains S-E.
               A TOS used as a store option on the line immediately preceding
               this instruction will assume this temporary name.

PADD   The pre-adder contents is placed in the R-BUS register.

PL   The Program Limit register, PL, is placed in the R-BUS register.

RA   The register named RA by the hardware namer is placed in the
               R-BUS register.

RB   The register named RB by the hardware namer is placed in the
               R-BUS register.

RBUS   The R-BUS register is unchanged.

RC   The register named RC by the hardware namer is placed in the
               R-BUS register.

RD   The register named RD by the hardware namer is placed in the
               R-BUS register.

SP∅   Scratch Pad ∅, SP∅, is placed in the R-BUS register.

SP1   Scratch Pad 1, SP1, is placed in the R-BUS register.

SR   The Stack Register counter, SR, is placed in the R-BUS
               register, preceded by 13 leading zeros.

UBUS   The output of the shifter or U-BUS is placed in the R-BUS
               register.

X   The index register, X, is placed in the R-BUS register.

XC   If the   index   bit of the current instruction is zero, then 0 is placed in the R-BUS register, otherwise the index register is placed in the R-BUS register.

Z   The stack limit Register is placed in the R-BUS register.

NOTE: Indirect bit = $(CIR(4) \cdot \overline{Mem\ Ref} + CIR(5) \cdot Mem\ Ref) \cdot \overline{CLIBFF}$

S-BUS Field

(Blank)   Zero is placed in the S-BUS register.

*CC   SBUS(8:9) := STATUS(6:7)
and if STATUS (6:7) = 00 then S-BUS(7) := 1
else S-BUS(7) := 0

All other bits of SBUS are zeroed.

CIR   The contents of current instruction register is placed in the S-BUS register.

— CPX1   Interrupt status register, active only when the micro-processor is in RUN mode, is placed in the S-BUS register.

— CPX2   Interrupt status register, active only when the micro-processor is in HALT mode, is placed in the S-BUS register.

*CTRH   S-BUS REG(4:9) <= CNTR(0:5). This will be used mostly in floating point exponent manipulations.

*CTRL   S-BUS REG(10:15) <= CNTR(0:5)

DB   The data base register, DB, is placed in the S-BUS register.

DL   The data limit register, DL, is placed in the S-BUS register.

IOA   The I/O address register is placed in the S-BUS register.
(Reads Interrupting Device NO).

IOD   The I/O data register is placed in the S-BUS register.
(Reads Direct Data Buffer).

MASK   The Mask register is placed in the S-BUS register.

MOD   A constant is brought to the S-BUS register in the following way:
S-BUS(0:4) <= 0;
S-BUS(5:7) <= Interrupting module number
S-BUS(8:15) <= If CPU1 then 4 else 8.

*Unless otherwise noted, remaining bits are zero.

— See explanation of interrupts.

OPND	The operand register is placed in the S-BUS register.
P	The Program counter, P, is placed in the S-BUS register.
PADD	The pre-adder contents is placed in the S-BUS register.
PB	The Program Base register, PB, is placed in the S-BUS register.
Q	The Stack Marker Pointer register, Q, is placed in the S-BUS register.
QDWN	It takes the lowest TOS register and put it in the S-BUS register in the following way: the TOS registers are renamed by NAMER + SR. RD is dispatched to the S-BUS and TOS registers are returned to their former names. A TOS register used in the STORE field of the previously executed instruction will assume a temporary name. A DCSR Special Option is needed to complete the Operation.
RA	The register named RA by the hardware namer is placed in the S-BUS register.
RB	The register named RB by the hardware namer is placed in the S-BUS register.
RC	The register named RD by the hardware namer is placed in the S-BUS register.
RD	The register named RD by the hardware namer is placed in the S-BUS register.
SBUS	The S-BUS register is unchanged.
SM	The memory Top of Stack pointer register, SM, is placed in the S-BUS register.
SP1	Scratch Pad register 1, SP1, is placed in the S-BUS register.
SP2	Scratch Pad register 2, SP2, is placed in the S-BUS register.

SP3   Scratch Pad register 3, SP3, is placed in the S-BUS register.

STA   The Status register, STA, is placed in the S-BUS register.

SWCH   The switch register contents is placed in the S-BUS register.

UBUS   The output of the shifter or U-BUS is placed in the S-BUS
            register.

Function Field

ADD   The contents of the R-BUS and the S-BUS registers are added and
            the result is left on the T-BUS.

ADDO   The contents of the R-BUS and the S-BUS are added and the
            result is left on the T-BUS. The overflow and carry bits
            are enabled. CCA is set from the U-BUS.

AND   The logical AND of the R-BUS and the S-BUS is left on the T-BUS.

TASL   Causes a 3 Register Arithmetic Shift left of the UBUS,
            SP1 and the R-BUS Register containing the most, middle, and
            least significant words, respectively. SL1 is required in the
            Shift field and the direction of the shift is left. The sign
            bit is preserved.

            T-BUS:=   SREG;
            UBUS(∅):=   TBUS(∅);
            UBUS(1:14):=   TBUS(2:15);
            UBUS(15):=   SP1(∅);
            SP1(∅:14):=   SP1(1:15);
            SP1(15):=   R REG(∅);
            RREG(∅:14):=   RREG(1:15);
            RREG(15):=   ∅;

TASR   Causes a 3 Register Arithmetic Shift Right of the UBUS, SP3,
            and the S-BUS Register containing the most, middle and least
            significant words respectively. SR1 is required in the shift
            field and the direction of the shift is Right. The sign bit
            is propagated.

            TBUS:=   RREG;
            UBUS(∅:1):=   TBUS(∅);
            UBUS(2:15):=   TBUS(1:14);
            SP3(∅):=   TBUS(15);
            SP3(1:15):=   SP3(∅:14);
            SREG(∅):=   SP3(15);
            SREG(1:15):=   SREG(∅:14);

BNDT   This operation performs a hardware bounds test of an address.
            T-BUS := R-BUS - S-BUS. If CARRY = 1, the next microinstruction
            is fetched. If CARRY = 0, and the machine is in user mode, a
            hardware micro-jump is made to address 2. BNDT takes precedence
            over the skip field.

CAD   The 1's complement of the S-BUS is added to the R-BUS and
                the result placed on the T-BUS.

CADO   Same as CAD with addition that the carry and overflow bits
                are enabled. CCA is set from the U-BUS.

CAND   T-BUS := R-BUS AND (S-BUS).

CRS   The T-BUS is circular shifted right(SR1) or left(SL1) one bit
                and put on the U-BUS. U(0) := T(15) if SR1, or U(15) := T(0)
                if SL1. Implied T-BUS := R-BUS + S-BUS.

CTSD   This function performs a double register shift of the T-BUS
                and a scratch pad register. A left shift, indicated by an
                SL1 in the shift field, expects the least significant word in
                SP1. A right shift(SR1) expects the least significant word in
                SP3. The type of shift is determined from the contents of the
                CIR as follows. T-BUS := R-BUS + S-BUS implied.
                        CIR(7) = 1   Circular shift
                        CIR(7:8) = 0,1   Logical shift
                        CIR(7:8) = 0,0   Arithmetic shift
                NOTE: Both SP1 and SP3 get shifted on CTSD.

CTSS   The T-BUS is shifted in a manner determined by CIR(7:8) as
                follows. Implied T-BUS := R-BUS + S-BUS.
                        CIR(7) = 1   Circular shift
                        CIR(7:8) = 0,1   Logical shift
                        CIR(7:8) = 0,0   Arithmetic shift
                NOTE: The direction is determined by shift field.

DVSB   This function performs the subtract, shift, and test necessary
                to implement a divide algorithm. To start, F2 = 0, the divisor
                is in the S-Reg., and the double word dividend is in the R-Reg.
                (MSW) and SP1. SL1 must be in the shift field. One bit quotient
                comes in SP1(15).

                DVSB ALGOL DEFINITION:
                        TBUS:= RBUS-SBUS;
                        UBUS(∅:14):= TBUS(1:15); <<BY SL1 in shift field>>
                        If ALU carry or F2=1 then
                            BEGIN
                                RREG(∅:14) := UBUS(∅:14);
                                RREG(15) := SP1(∅);
                                SP1(∅:14) := SP1(1:15);
                                SP1(15) := 1;
                                F2 := TBUS (∅);

```
                                    END
                        else
                            BEGIN
                              RREG(∅:14) := RREG(1:15);
                              RREG(15) := SP1(∅);
                              SP1(∅:14) := SP1(1:15);
                              SP1(15) := ∅;
                              F2 := RREG (∅);
                            end;
```

INC   T-BUS := R-BUS + S-BUS + 1

INCO   Same as INC with the addition that the carry and overflow
               bits are enabled. CCA is set from the U-BUS.

IOR   The R-BUS and S-BUS are logically ORed together and the result
               placed on the T-BUS.

JMP   This function performs a micro-jump to the ROM address specified
               in bits 20 to 31 if the condition contained in the skip field
               is met. If the skip condition is not met, the next ROM instruc-
               tion in sequence is fetched. Also, implied U-BUS := T-BUS := S-BUS.
               Execution requires two clock cycles for non-data-dependent SKIPS
               and 3 cycles for all data dependent SKIPS.

JSB   This function causes a subroutine jump, and is executed like
               the JMP function with the following addition. The RAR register
               is stored into the SAVE register before the target address is
               transferred from the ROR. The return address is saved in the
               SAVE reg. Since there is 1 SAVE, subroutines may be nested only
               1 deep.

MPAD   This function performs the shift, test, and add functions
               necessary to implement a multiply algorithm. To start, the
               multiplier is in SP3, the multiplicand is in the R-BUS register,
               and the S-BUS register = 0. An SR1 is required in the shift field.
               One bit result comes in SP3(0).
                        T-BUS := R-REG + S-REG; U-BUS(1:15) := T-BUS(0:14)
                        U-BUS(0) := ALU carry; if SP3(15) = 1, then S-Reg :=
                        U-BUS, SP3(1:15) := SP3(0:14); SP3(0) := T(15); else
                        S-REG(1:15) :=·S-REG(0:14),SP3(1:15) := SP3(0:14),
                        SP3(0) := S-REG(15).
```

PNLR            A control panel function.  The appropriate register selected
                from the front panel is brought to the T-BUS through ALU.
                This is executed out of ROR2 giving garbage on the first cycle.


PNLS            A control panel function.  The U-BUS is stored in the appropriate
                register selected from the front panel.

                NOTE:  Store and SKIP field must be NOPS.


The following two functions are repeat commands and operate in the following
manner.  The microinstruction following the repeat command is executed over and
over until the skip field condition of the repeated instruction is met.  The
instruction is then terminated and normal microprocessing proceeds.  The skip
field of the repeat instruction may not be used, except as shown below.  The
two repeat functions differ only in what they do during their execution, not
in the operation of the repeated instruction.


REPC            Normal repeat function that has implied T-BUS := R-BUS + S-BUS.


REPN            Send skip field contents to CNTR, CNTR(0) = 1, and implied T-BUS
                := R-BUS + S-BUS.


ROM             Bits 16-31 of this instruction are placed in the R-BUS reg.
                Implied T-BUS := R-BUS + S-BUS.  Note that immediate operand
                voids the shift, R, special, and skip fields.


ROMI            Same as ROM except implied inclusive - OR of R and S BUS.


ROMN            This function is like ROM except implied T-BUS := R-BUS
                AND S-BUS.


ROMX            This function is like ROM except implied T-BUS := R-BUS XOR S-BUS.


RPTY            T-BUS := R-BUS + S-BUS.  This function reverses parity generation
                for diagnostic use only.


SUB             T-BUS := R-BUS - S-BUS.

SUBO               Like SUB, except carry and overflow bits enabled. CCA is set from the U-BUS.

UBNT               Unconditional bounds test. T-BUS := R-BUS - S-BUS. If no carry, than a hardware jump to ROM ADDRESS 2.

XOR               T-BUS := R-BUS EXCLUSIVE OR S-BUS.

Shift Field


(blank)          No shift, U-BUS := T-BUS.


LLZ              "Left to left and zero" places the left byte of the
                 T-BUS in the left byte of the U-BUS and places zeros
                 in the right byte of the U-BUS.


LRZ              "Left to right and zero" places the left byte of the T-BUS
                 in the right byte of the U-BUS and places zeros in the left
                 byte of the U-BUS.


RLZ              "Right to left and zero" places the right byte of the T-BUS
                 in the left byte of the T-BUS and places zeros on the right
                 byte of the U-BUS.


ROT              "Rotate" places the right byte of the T-BUS in the left byte
                 of the U-BUS and the left byte of the T-BUS in the right in the
                 the right byte of the U-BUS.


RRZ              "Right to right and zero" places the right byte of the T-BUS in the
                 right byte of the U-BUS and places zeros in the left byte of the
                 U-BUS.


SL1              "Shift left one" shifts the T-BUS one bit left onto the U-BUS.
                 When used with TASL, CTSS, CRS, CTSD and DVSB in the function
                 field, refer to those descriptions to determine the action taken.
                 This option may be used alone to perform a single logical shift
                 where zero is brought in and bit $\emptyset$ is lost.


SR1              "Shift right one" shifts the T-BUS on right onto the U-BUS.
                 When used with TASR, CTSS, CRS, CTSD and MPAD in the function
                 field, refer to those descriptions to determine the action
                 taken.  This option may be used alone to perform a single
                 logical right shift where zero is brought in and bit 15 is lost.

Store Field

(blank)        No store.

BSPØ           Stores U-BUS into the COR and into SPØ.  It disables the
               special field and enables the MCU options, one of which must
               be used.  It issues a LO Request.

BUSH           Stores U-BUS in COR (CPU Output Register) and makes a BUS-HI
               request.  It disables the special field and enables the MCU options,
               one of which must be used.

BUSL           COR := U-BUS.  Initiates a low request for the bus.  It disables
               the special field and enables the MCU options, one of which must
               be used.

CTRH           Counter high stores U-BUS(4:9) in the counter.

CTRL           Counter low stores U-BUS(10:15) in the counter.

DATA           Stores the U-BUS into the COR and issues a high request and
               transfers data to the module last addressed by the "TO" register.

DB             Stores the U-BUS in the Data Base Register, DB.

DL             Stores the U-BUS in the Data Limit register, DL.

* IOA          Stores the S-BUS into the I/O Address register.

* IOD          Stores the S-BUS into the I/O Data register.

* MASK         Stores the S-BUS in the Mask register.

MREG           The contents of Namer is added to two bits SP1(14:15) to obtain
               temporary name.  This is used to reference a memory element that
               happens to lie in the TOS registers.  SP1(14:15) contains S-E.
               TOS registers used in the R and S fields following this instruction
               will assume the temporary name.

  *  NOTE:  A CF1 special option must be executed after the occurrence of these
            options to restore F1 to CPU use.  The data is taken from the S-BUS
            option on the instruction following the I/O store option.

| | |
|---|---|
| P | Stores the U-BUS into the program counter, P. |
| PB | Stores the U-BUS into the Program Base register, PB. |
| PL | Stores the U-BUS into the Program Limit register, PL. |
| PUSH | Stores the U-BUS into the RD register, increments the SR register by one and at the end of the microinstruction cycle renames the TOS registers such that: N(RA) := N(RB) : RC, N(RC) := RD, N(RD) := RA. |
| Q | Stores the U-BUS in the Stack Marker Pointer, Q. |
| QUP | The TOS registers are renamed by NAMER + SR. Temporarily named RA := U-BUS. The TOS register names are returned to NAMER - however incrementing of SR is not implicit. INSR (inc. SR) must appear in the special field in order to increment SR. TOS registers used in the R and S fields following this instruction will assume the temporary name. |
| RA | Stores the U-BUS in the register named RA at the beginning of the cycle. |
| RAR | Stores U-BUS(4:15) in the RAR(0:11). This takes 3 cycles. |
| RB | Stores the U-BUS in the Register named RB at the beginning of the cycle. |
| RC | Stores the U-BUS in the Register named RC at the beginning of the cycle. |
| RD | Stores the U-BUS in the Register named RD at the beginning of the cycle. |
| SM | Stores the U-BUS into the memory stack pointer, SM. |
| SP∅ | Stores the U-BUS into scratch pad register ∅, SP∅. |

SP1    Stores the U-BUS into scratch pad register 1, SP1.

SP2    Stores the U-BUS into scratch pad register 2, SP2.

SP3    Stores the U-BUS into scratch pad register 3, SP3.

STA    Stores the U-BUS into the Status Register.

X     Stores the U-BUS into the index register, X.

Z     Stores the U-BUS into the stack limit pointer, Z.

Special Field

(blank)        No special option.


CCA            Sets the condition code bits in the status word to
               CCL if T-BUS < 0.
               CCE if T-BUS = 0.
               CCG if T-BUS > 0.


CCE            Sets the condition code bits in the status word to CCE.


CCG            Sets the condition code bits in the status word to CCG.


CCL            Sets condition code bits in status word to CCL.


CCPX           Clears the interrupt status register bits as specified by the
               true bits on the U-BUS.  (See explanation of interrupts.)


CCRY           Clear the carry bit in the status word.


CCZ            Sets condition code bits in status word to CCE if T-BUS = 0
               and CCG if T-BUS not equal Ø.


CF1            At the.end of the cycle, CF1 clears Flag 1.


CF2            At the end of the cycle, CF2 clears Flag 2.


CF3            At the end of the cycle, CF3 clears Flag 3.


CLIB           At the end of the cycle, CLIB clears the indirect line until
               a NEXT option is encountered.


CLO            At the end of the cycle, CLO clears the overflow bit in the
               status word.

| | |
|---|---|
| CTF | Stores the ALU carry in Flag 1 at the end of the cycle. |
| DCSR | Decrements the SR counter by 1. |
| INCN | Increments the Namer RA ← RD, RB ← RA, RC ← RB, RD ← RC. |
| FHB | U-BUS(0) := FLÁG1 |
| HALT | Enables the front panel by setting CPX(0) := 0. |
| HBF | FLAG1 := U-BUS(0) |
| INCT | Increments the counter by 1 (modulo 64). |
| INSR | Increment SR by 1. |
| LBF | Low bit to flag 2.  F2 := U-BUS(15). |
| POP | This option decrements the SR by 1 and then renames the TOS registers such that:<br>N(RA) := RD, N(RB) := RA, N(RC) := RB, N(RD) := RC. |
| POPA | Exactly like POP, except CCA is set on the contents of the U-BUS. |
| SCRY | Set the carry bit in the status word. |
| SDFG | Sets the dispatcher flag CPX2(12) := 1. |
| SF1 | Sets flag 1 at the end of the cycle. |
| SF2 | Sets flag 2 at the end of the cycle. |

SIFG            Sets the interrupt flag CPX2(11) := 1.

SOV             Sets the overflow bit in the status word at the end of the cycle.

SR∅             Sets SR to zero, during the cycle.  Note:  No other SR operation
                during the cycle is allowed.

CCB             Sets CCB on contents of SP1(8:15):
                        CCL = Special
                        CCE = Alphabetic
                        CCG = Numeric


NOTE:   CCL = STA (6:7) = ∅1
        CCE = STA (6:7) = 1∅
        CCG = STA (6:7) = ∅∅

Skip Field

The skip field does one of two things :

1. Sets the condition met flag or

2. Initiates a hardware micro-jump.  A hardware micro-jump needs no jump
   target in the micro-instruction.

The condition met flag after a REPN or REPC function option indicates
the condition on which to terminate the repeated micro-instruction.  Otherwise
it indicates that the next micro-instruction is to be skipped.

| | |
|---|---|
| (blank) . | No skip option |
| ➡BIT6 | Condition met if bit 6 of the U-BUS is a 1. |
| ➡BIT8 | Condition met if bit 8 of U-BUS is a 1 |
| ➡CRRY | Condition met if the carry out of the ALU is a one. (Note:  this is <u>not</u> the carry bit in the status word.) |
| CTRM | Condition met if the counter is all ones. (NOTE:  When INCT CTRM options occur the counter is tested before it is incremented.) |
| ➡EVEN | Condition met if the U-BUS(15) = 0. |
| F1 | Condition met if at the beginning of the cycle, flag 1 is set. |
| F2 | Condition met if at the beginning of the cycle, Flag 2 is set. |
| F3 · | Condition met if flag 3 is set. |
| *INDR | Condition met if the indirect bit of the current instruction register is set. |
| JLUI | Conditional hardware microjump to the address that the lookup table is displaying if the indirect bit of the CIR is not set. CLIB must be used to guarantee a jump on all instructions. |

*INDR= (CIR(4)·$\overline{\text{MEM REF}}$ + CIR(5) · MEM REF) ·$\overline{\text{CLIBFF}}$

**NCRY**  Condition met if the carryout of the ALU is zero.
(Note this is <u>not</u> the carry bit in the status word.)

**NEG**  Condition met if U-BUS(0) = 1.

**NEXT**  This function causes the hardware to get the next user
instruction and decode it.  If stackop A has just been
executed and stackop B is not a NOP, then the hardware
executes stackop B.  Any other instructions in the CIR
causes COR := P and an RWN to be dispatched to memory.
P := P + 1.  CIR := NIR, and NIR awaits the data from
memory.  CIR is then decoded, the preadder adjusted and
sent to the R-BUS register.  This action primes the pipe
for the first microinstruction fetch.  Implies R-BUS +
S-BUS during the first cycle.  The S-BUS register con-
tains a Base register in memory reference instructions.

**NF1**  Condition met if at the beginning of the cycle, flag 1 is
cleared.

**NF2**  Condition met if at the beginning of the cycle, flag 2
is cleared.

**NPRV**  Condition met if at the beginning of the cycle the
privileged mode bit is not set.

**NSME**  Condition met if all the bits of the T-BUS are not the same.

**NZRO**  Condition met if the T-BUS is non-zero.

**ODD**  Condition met if the U-BUS(15) = 1.

**NOFL**  Condition met if overflow out of the ALU does not occur.
(Note this is <u>not</u> the overflow bit in the status word.)

**POS**  Condition met if the U-BUS(0) = 0.

**RSB**  Hardware micro-jump to the address held in the SAVE
register.  The SAVE register contents is transferred to
the RAR register.

SR4            Condition met if the SR register is 4.

SRL2           Condition met if the SR register is less than 2.

SRL3           Condition met if the SR register is less than 3.

SRN4           Condition met if the SR register is not 4.

SRNZ           Condition met is the SR register is non-zero.

SRZ            Condition met if the SR register is zero.

TEST           Condition met if any interrupt is pending.

UNC            Condition met unconditionally.

— ZERO         Condition met if the T-BUS is zero.


— Note:  These options, when used with JMP or JSB functions, cause a 1

        cycle freeze.

MCU OPTIONS (override special field if there is 'BUSH', 'BSPO' or
'BUSL' in store field).


CMD                    MCU "to" lines and command Bus are taken from the CRL
                       Register on the Bus transfer.


CRL                    Takes the MCU bus and stores it in the CRL register.  It is
                       used as BUSH CRL.


CWA                    Clear write address initiates a bus request for a memory
                       and tells the memory that the data on the bus is an address,
                       and that the data will be sent on the next transmission.
                       Memory will hold the address in the memory address register.
                       The memory will be busy for all modules until the data is
                       received via the DATA using a high bus request.


NIR                    'Next instruction Register'.  NEXT in the skip field brings
                       the new instruction in NIR.


OPND                   Takes information from MCU bus and puts it in OPND register
                       (used as BUSH OPND).


RNWA                   Read no write address initiates a bus request for a memory
                       and tells the memory that the data on the bus is an address
                       the contents of which is to be sent back to the OPND
                       register and not to complete the write cycle leaving all
                       ones in the word read.


RWA                    Read write address is the same as RNWA except that the original
                       contents of the cell are restored.


RWAN                   Read Write Address and Next Instruction is the same as
                       RWA except the results are returned to both the OPND and
                       the NIR registers.


RWN                    Like RWA with the exception that instead of sending the memory
                       cell contents to OPND·register,. it sends it in NIR (next in-
                       struction register).